

# Agentic RAG Chatbot for Multi-Format Document QA

- Powered by Model Context Protocol (MCP)
- Author: RAGUNATH R
- Date: July 2025

# Problem Statement & Objective

- Problem: Build a chatbot to answer questions from diverse document formats
- Goal: Enable intelligent document Q&A with modular agents and message-driven architecture
- Supported Formats: PDF, PPTX, CSV, DOCX, TXT/Markdown

# Architecture Overview (Agentic + MCP)

## Agents:

- IngestionAgent: Parses and splits documents
- RetrievalAgent: Embeds and searches chunks
- LLMResponseAgent: Generates answers from context
- CoordinatorAgent: Manages flow & tracks context

## MCP Messaging:

- Structured messages for communication between agents
- Fields: sender, receiver, type, trace\_id, payload

# System Flow with MCP Message Passing

- 1. Upload → Coordinator:  
INGESTION\_REQUEST
- 2. Coordinator → IngestionAgent →  
RetrievalAgent → LLMResponseAgent
- 3. Response streamed back to UI with sources
- MCP Message: {type, sender, receiver,  
trace\_id, payload}

# Tech Stack Used

- Frontend: Gradio (with custom CSS)
- LLM: Llama 3.1 8B via HuggingFace Inference cli
- Vector Store: FAISS
- Embeddings: sentence-transformers/all-MiniLM-L6-v2
- Parsers: PyPDF2, python-docx, python-pptx, pandas
- Messaging: In-memory bus using Python classes

# DocAgent-Demo

The screenshot shows the DocAgent web application running on Hugging Face Spaces. The browser address bar displays `huggingface.co/spaces/ragunath-ravi/DocAgent`. The application header includes navigation links for Spaces, ragunath-ravi, DocAgent, like, Running, Logs, App, Files, Community, and Settings.

The main interface is titled "Agentic RAG Assistant" with the subtitle "Upload documents and ask questions - powered by Multi-Agent Architecture".

On the left sidebar, there is an "Upload Documents" section showing a file named "Agentic\_RAG\_Presentation.pptx" (32.5 KB). Below this, a green message states "Documents processed successfully! You can now ask questions." and a large orange button labeled "Process Documents".

The central chat area shows a conversation with the assistant. The user's input is "according to the documents what they are telling". The assistant's response is "According to the documents, Agentic RAG Chatbot for Multi-Format Document QA is a chatbot built to answer questions from diverse document formats." Below this, the user asks "what are the type of document formats ae allowed".

At the bottom, there is a text input field with the placeholder "Ask about your documents..." and a "Send" button. Below the input field, there are four example questions: "What are the main topics discussed?", "Summarize the key findings", "What metrics are mentioned?", and "What are the recommendations?".

The footer of the application includes links for "Use via API", "Built with Gradio", and "Settings".

# UI Features & Challenges

- Multi-file upload, multi-turn chat, visible examples
- Dark theme UI with styled messages and input box
- Challenges: LLM streaming consistency, parser errors, message delays
- Future: REST-based MCP, memory agent, GPU optimization